Dealing with Hardware-related Disturbances in Organic Computing Systems

Markus Görlich-Bucher[1]

Abstract: The ability to withstand disturbances while remaining functioning in a desired way is regarded as a crucial element in the field of Organic Computing. However, current approaches to *self-healing* and *robustness* fail in considering hardware-related breakdowns. These disturbances differ from software-sided disturbances in various aspects: They persist until being repaired, therefore their removal necessitates maintenance actions performed by human repair workers. Furthermore, they may be predicted to a certain degree. In this article, we formulate a problem statement and various requirements an OC system must fulfil in order to increase its robustness against hardware-related disturbances. Furthermore, we present a working plan for a PhD project concerning the investigation of several aspects of the previously motivated problem statement.

Keywords: Organic Computing; Robustness; Resilience; Self-Healing; Predictive Maintenance

1 Introduction

Over the last decades, *Organic Computing* (OC) [MST17b] and its principles have been developed in order to reduce the increasing complexity in information and communication technology by moving design-time decisions to runtime. In order to achieve these goals, OC systems are conceived to adapt and organize themselves in an autonomous way. Therefore, they incorporate various so-called *self-x* properties, such as self-organization, self-configuration or self-healing. The concepts developed in the scope of OC have been employed and tested in various application scenarios in the past years. Considerable examples include traffic control [Pr11] or self-organising communication systems [Zi11]. A crucial element to allow such autonomy is the ability to be *robust* against external or internal disturbances, therefore, to maintain (or at least return to) a desired *system state* despite the occurrence of failures. Intuitively, mechanisms developed to increase a system's robustness can be summarized under the self-healing capabilities of OC. An aspect that distinguishes OC from other approaches in the field of autonomous systems, such as *Autonomic Computing* [LMD13], is the fact that OC systems are expected to be able to interact with their surrounding environment. Therefore, OC systems are supposed to be equipped with sensors and actuators. Although conceived to solve complex real-world problems necessitating a notable amount of sensors and actuators, existing self-healing

[1] Organic Computing Group, University of Augsburg, Germany markus.goerlich-bucher@informatik.uni-augsburg.de

doi:10.18420/inf2019_ws55

mechanisms in OC mostly focus on software-related disturbances. Moreover, no general-purpose methodology on how to deal with hardware-sided disturbances can be found in the literature. These *Physical Disturbances*, as they are called in the following, differ from software-sided disturbances in various aspects: A damaged sensor or actuator remains damaged, therefore, a physical disturbance does not end by itself. Besides, no control mechanism can end them, at least, it might be able to compensate them using e.g. redundant infrastructure. Furthermore, a physical disturbance requires human intervention in form of *maintenance activities* to handle, respectively end it.

Maintenance is a well-discussed aspect in various other scientific disciplines, such as manufacturing or economic sciences. Hereby, current developments mostly focus on an approach called *Predictive Maintenance* (PdM) [Se17]: Sensor data or observations are used to predict machine failures or ongoing degradation using statistical approaches or machine learning. Using these predictions, defective machines (or parts of them) can be maintained or replaced in a proactive manner, reducing the downtime of the corresponding machine (and therefore the costs) to the maintenance action itself. Furthermore, in more complex scenarios, probably with dependencies between machines or components, the gathered predictions can be used to construct maintenance schedules [vP13]. Depending on the application scenario, these schedules may be optimized for lower costs or shorter downtimes.

In this paper, we present a preliminary concept for investigating various aspects of the integration of maintenance activities into the scope of OC systems. At first, we investigate the status quo of robustness, resilience and self-healing in the context of OC (Section 2). Based on the previously explained characteristics of physical disturbances we formulate a problem statement in section 3. Hereby, we focus on how physical disturbances might affect OC systems, how they differ from software-related disturbances, and conclude various requirements an OC system should fulfil in order to increase its robustness to them. Afterwards, we suggest a novel evaluation measure as well as a formalization in form of a *Mixed Observability Markov Decision Process* that can be used to determine the long-term effects of physical disturbances on the robustness of an OC system (Section 4). Afterwards, we suggest a working plan for a PhD project concerned with investigating various aspects of identifying disturbances, predicting their possible influence on the overall robustness of a system, as well as methodologies to integrate human-dependent maintenance activities into the scope of OC (Section 5).

2 Related Work

2.1 Robustness, Resilience, and Self-Healing in OC: Status Quo

The IEEE standard glossary of software engineering terminology defines robustness as "The degree to which a system or component can function correctly in the presence of

invalid inputs or stressful environmental conditions" [IE90]. In terms of OC, robustness is perceived as the ability of a system S to recover from any kind of previously known disturbance. Formally defined in [Sc10], a system state $\vec{z}_t$ (describing the state of S at any given time t) is mapped with respect to some evaluation criteria $\mu(\vec{z}_t)$ into various *state subspaces*. States mapped into the *target space* denote ideal states, while states mapped into the *acceptance space* denote acceptable, but not ideal states. Besides, a mapping into the *survival space* denotes states from which the *control mechanism* (CM) of the system is able to recover at least to the acceptance space, while mappings into the *dead space* denote unrecoverable states. An occurrence of a disturbance $\omega \in D$ with D denoting a set containing all possible disturbances changes the system state $\vec{z}_t$ to a new state $z_{t+1} = \omega(\vec{z}_t)$. The actual robustness of S is measured depending on the state space $\vec{z}_t$ is moved into by any disturbance in D. For example, a system that needs to deal with a disturbance that changes $\vec{z}_t$ into the dead space is considered as not robust at all. The concept of robustness was later on revised towards an approach called *Quantitative Robustness* [To18]. Here, the focus lies on measuring a systems robustness as well as the influence of disturbances in a comparable manner by using some application specific utility measure U. Both the strength and the duration of a disturbance is measured by observing the change of U over time. The actual state space boundaries are represented by utility limits, e.g. θ_{target} the upper boundary of the target space. The actual robustness of a system is determined by the change of U during and after the occurrence of a disturbance. For example, a recovery to $U \geq \theta_{acc}$ during a disturbance and to $U \geq \theta_{\text{target}}$ after the end of a disturbance denotes a strongly robust system.

Another concept related to robustness was developed under the term *resilience*, defined as "pro-active robustness" in [ST16]. Here, the focus lies on predicting upcoming incidents that can affect a systems robustness by utilizing suitable machine learning techniques within the CM. The concept was used to predict traffic flows in [STH16].

Robustness and Resilience provide measures (or at least concepts) to evaluate the influence of disturbances on an OC system. However, an actual compensation or healing might incorporate other concepts and techniques. As mentioned before, research towards increasing a systems self-healing capabilities mostly focuses on healing software-sided disturbances (cf. e.g., [Sc11], [Na09]). Existing approaches concerned with healing physical disturbances mostly focus on redundant hardware. A notable example is the six-legged robot OSCAR that compensates failing legs by utilizing the remaining, functioning legs [Ma11]. Another recent example lies in autonomous sensor networks, where failing sensors are compensated by other sensors in their neighbourhood [Ja14]). In general, the term "self-healing" is discussed in various other scientific disciplines as well. A comprehensive survey on the topic can be found in [Fr13].

2.2 (Predictive) Maintenance

As mentioned before, PdM (also termed *Condition-based Maintenance*) - or maintenance in general - is well-discussed in various other scientific disciplines. A review of various prognostic models used in this field can be found in [PDZ10]. In [AK12], an overview focussing on the actual applicability in industrial applications is provided. Another review evaluates the availability of contemporary sensor technology and wireless networks for the field of PdM [Se17]. In the last years, plenty of work on predicting failures using different kinds of machine learning techniques has been published. Prominent examples include artificial neural networks (cf. eg., [Fu04], [Wu07], [GWJ17]), Genetic Algorithms (cf. eg., [BDR13] [LVT16]), as well as unsupervised learning techniques, such as Clustering or Anomaly Detection (cf. eg., [KS17], [Kr14]).

Another related aspect in the field of maintenance lies in the scheduling of actual maintenance activities. Notable work focuses on considering economic, structural and stochastic dependencies between components within manufacturing plants [vP13] as well as the influences between the structure of complex, multi-component systems and the deterioration of actual single components [NDG14].

3 Problem Statement

As already addressed in the introduction, physical disturbances differ from software-sided disturbances in various aspects. In the following, we explain the differences and potential problems they yield concerning the design of OC systems in detail.

First of all, the healing of a physical disturbance will most likely not lie in the scope of the OC system itself. Although concepts like *self-repair* - for example, using drones - have been investigated in the scientific community extensively for various years now, we assume that we won't see, for example, fully autonomously, self-repairing manufacturing plants in the next couple of years. Therefore, the healing of physical disturbances will necessitate humans for now. At this point, one could argue that a human repair worker does not fit into the concept of "self-healing", as it is clearly not the system itself that does the job. However, we suggest to take the human out of the scope of the maintenance- and repair-actions as far as possible: It appears suitable to degrade humans to tools the OC system utilizes to heal itself. Therefore, humans can be regarded as (more or less trustworthy) agents in the scope of the system. They might have their own desires and goals, but in the end, they are expected to execute the work the CM delegates to them.

In order to reduce human intervention as far as possible, the CM must be able to identify the source of the physical disturbance, therefore, be aware of the structure of the underlying system. Furthermore, it must be able to determine a suitable time frame for both proactive as well as reactive maintenance actions. Finally, a physical disturbance does not necessarily lead do a decrease of the systems' utility: In the example of the six-feet robot OSCAR

mentioned in the previous section, the CM is able to compensate a failing leg using the other legs. While this redundancy guarantees a robust system in the first place, it is more vulnerable to additional physical disturbances afterwards.

In conclusion, we formulate three requirements to an OC system in order to be robust to physical disturbances:

1. Depending on the hardware of the underlying system, physical disturbances might be predicted to a certain degree. In order to assign proactive maintenance actions that can avoid time consuming breakdowns, the CM must be able to predict disturbances in advance, whenever possible. These predictions should take the current system configuration, as well as possible future changes of the configuration into account.
2. Physical disturbances cannot be ended by a CM. They persist until repaired by a maintenance action, presumably executed by a human worker. In order to ensure the autonomy of an OC system, the responsibility and the powers of human workers should be kept as limited as possible. Therefore, the identification of the source of a physical disturbance, as well as the decision on and assignment of necessary maintenance actions must be performed by the CM. In order to fulfil this requirement, the CM must be aware of the structure of its underlying system to a certain degree.
3. The effect of physical disturbances on the overall robustness of a system might depend on the redundancy of the used hardware. Accordingly, the CM must be able to estimate the influence of possible future physical disturbances on the systems' utility at any time in order to assess if there exist situations in which it won't remain robust. The CM must be able to construct a suitable maintenance schedule depending on the actual disturbance predictions, the system configuration, the availability of human workers as well as possible combinations of multiple future physical disturbances.

4 Preliminary Concept

As shown in the previous section, we assume that a physical disturbance does not necessarily lead to a change in a system's utility. Rather, it might be possible that the utility does not change until a certain combination of different physical disturbances appear at once (or one after another). In such scenarios, using the utility as a measure to evaluate the ability of a system to remain robust against possible future disturbances appears insufficient. Accordingly, it is of interest to evaluate and measure the effects of physical disturbances on hardware components of the system independently from their influence on the utility. As introduced in [GSH19], we evaluate the physical state of hardware components in form of an *Integrity* measure: A physical disturbance that affects (therefore damages) a component of the system changes its integrity. Furthermore, the overall integrity of all components of the system can be used to determine the robustness of the system. In simple words, the integrity measurement acts as an abstraction between physical disturbances and their

long-term effect on a system' utility. The concept becomes more clear when considering the following formalization: There exists an integrity measurement $\vec{i_t} \in I$ for each timestep t, where $\vec{i_t} = (i_t^{(1)}, i_t^{(2)}, \ldots, i_t^{(n)})$ with n denoting the number of components in the system. A binary description with 0 describing fully damaged components and 1 describing fully functioning components appears suitable for rather simple applications scenarios. However, a more sophisticated description will be necessary in case of more complex situations, such as hierarchical systems or components that remain functioning to a certain degree when affected by a physical disturbance. The influence of a physical disturbance $\delta \in D_{\text{physical}} \subseteq D$ and $\vec{i} \in I$ can be formalized as a probabilistic function $f := P(\vec{i}_{t+1}|\vec{i}_t, \delta)$ (called integrity function in the following). Accordingly, a physical disturbance may change the integrity of one or more components in $\vec{i_t}$. The influence of an integrity measurement on a system's utility U_t can now be expressed by $g := P(U_{t+1}|\vec{z}_t, \vec{i}_t)$. If f and g were fully known, we could determine the overall robustness of S against any disturbance in D_{physical}.

5 Research Plan

In the following, we present a brief research plan for a PhD thesis based on the previously motivated problem statement as well as our preliminary concept.

5.1 Prediction of Physical Disturbances

For now, we assume two different settings for predicting physical disturbances in OC systems: Components that already have the ability to predict failures among themselves (e.g. in manufacturing plants equipped with PdM capabilities), as well as components for which the CM makes predictions using observations. Besides, it is conceivable that a system might incorporate both types of components. The challenge among CM-based predictions is to choose learning paradigms that are able to learn to predict failures based on observations in a more or less automated manner: Following the OC-ish concept of moving design-time decisions to runtime, we can assume that no pre-labeled training data is available in advance. Though, we expect that the integrity measure for a given component is known at all times (e.g. by observing the system utility or designated sensors that report the state of a component), therefore, the use of *Reinforcement Learning* (RL)-techniques, using the integrity measurements as rewards, appears feasible. Hence, one goal of this part of the research plan lies in evaluating RL-algorithms regarding their applicability to predict physical disturbances. The challenge among prediction algorithms located within the actual underlying system lies in assessing their reliability: From the CM's point of view, they can be regarded as a black box.

5.2 Architectural Aspects and Integrity Measurement

Besides the previously mentioned differentiation between predictions in the CM and in the underlying system, various other architectural aspects need to be considered. The most prominent CM-concept in OC is the *Generic Observer/Controller*-Architecture (O/C) [To11], depicted in the first schema in Figure 1. In short terms, it consists of two components, namely observer and controller. The observer investigates the underlying *System under Observation and Control* (SuOC) and provides an evaluation of the current system state to the controller, which thereupon decides if control actions are necessary. Commonly, this control logic is represented by simple *IF-THEN* rules, for example implemented as an algorithm from the field of *Learning Classifier Systems* (LCS) [To11]. Depending on the application scenario, an OC system is not necessarily limited to a single O/C instance. As depicted in the second and third schema in Figure 1, distributed and hierarchical topologies are also possible, for example by equipping each component in a system with its own O/C instance, all of them encapsulated by another O/C instance.

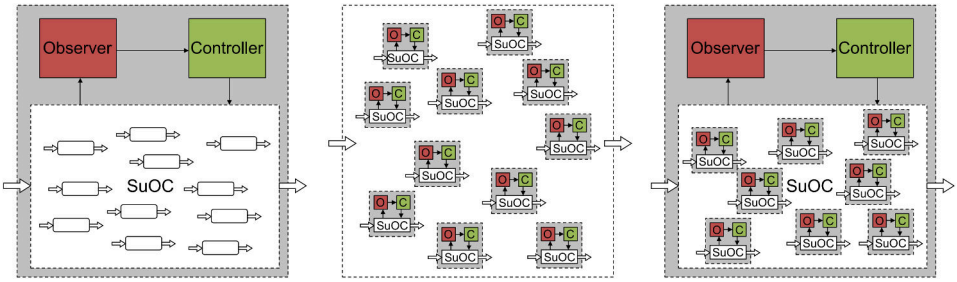

Fig. 1: Schematics of different O/C-topologies [To11]

Later on, the concept of the generic O/C architecture was enhanced to a so-called *Multi-Level Observer-Controller*-Architecture (MLOC) [MST17a]. Hereby, the first layer is intended to observe and control the actual SuOC, while the second layer focuses on generating new rules for the first layer's controller in cases where no suitable rule was found for an upcoming observation. In order to generate useful rules, a simulation of the SuOC might be part of the second layer. The third layer is used for communication and collaboration with other MLOC-instances, e.g. for exchanging knowledge from the local rulesets.

We aim to integrate a designated *Integrity-Component* (IC) into all layers of the MLOC-Architecture. Located in the observer on layer 1, the IC will be responsible for the identification of physical disturbances, the evaluation of their influence on the system's overall robustness, their prediction (if possible) as well as the calculation of a suitable integrity measurement. The determination of a proper maintenance schedule, however, is not up to the IC. Maintenance activities are considered as control actions, which lie in the scope of the controller. Therefore, the Observer is expected to report the integrity measurement as well as the predictions to the controller, where a proper schedule can be calculated. The layer 2 part of the IC, also located in the observer, is intended to use a simulation of

the SuOC to determine which integrity measures influence the robustness of the system, hence, to learn the integrity function. However, the applicability of such a scenario depends on the application scenario itself: While a simulation of a manufacturing scenario can be used rather easily to determine critical components, more complex scenarios might not be simulated at all (or not to a degree necessary to determine a correct integrity function). Finally, the IC in layer 3 is expected to communicate integrity measurements or learned predictors to other MLOC-instances that incorporate a similar SuOC.

Besides, different compositions of MLOC architectures might necessitate more complex integrity measurements (as already stated in the previous section): In a hierarchical or distributed topology, each single MLOC instance needs to decide which integrity measures of the underlying SuOC are relevant for other instances, and which ones can be kept private. Previous work that appears relevant in this context lies in the field of *computational self-awareness* [Le11].

5.3 Applications in different O/C-Architectures

The problem of identifying a suitable maintenance schedule in OC systems involves both certain information (The current system state and integrity, assuming that the corresponding observations are always correct), as well as uncertain information (disturbance prediction conducted by the observer or by an internal prediction algorithm in the SuOC, or unknown future breakdowns in general). Furthermore, it incorporates multiple control actions that might depend on limited *Resources* (e.g. human repair workers). This problem can be formalized as *Mixed Observability Markov Decision Process* (MOMDP) [On10]. We presented a rather minimal MOMDP that can be used to describe the prior problem in single O/C-architectures in [GSH19]. However, more complex MOMDPs will be necessary to describe hierarchical or distributed architectures. Furthermore, the design of such a MOMDP may vary depending on the type of desired maintenance schedule, e.g. a focus on a higher utility or lower downtimes of single components. Accordingly, in this part of the research plan, we aim to define MOMDPs for different O/C application scenarios.

Afterwards, suitable algorithms to solve the resulting MOMDPs need to be developed. Hereby, we focus on both learning paradigms that have been applied to Markov Decision Processes (MDPs) in OC before (cf. [St17]), as well as other machine learning concepts that have been applied successfully to MOMDPs or at least in the broader field of MDPs (cf. [On10]). As mentioned in section 2, plenty of work on how to schedule maintenance activities exists in various fields of research. Therefore, another part of the research plan focuses on conducting a comprehensive survey of existing maintenance scheduling concepts regarding their applicability in the field of OC, respectively to the designed MOMDPs.

5.4 Evaluatory Aspects

Physical disturbances may occur in any OC system - even if the SuOC does not incorporate actuators or sensors, the physical hardware on which the system relies might be exposed to such disturbances. The evaluation environment for an OC system that is robust to physical disturbances depends significantly on the type of actuators or sensors the system is using and which abstraction level of the whole architecture we need to evaluate. In the following, we suggest various approaches that can be used to build a suitable evaluation environment for our proposed concept.

Simulation Environment An intuitive application scenario for the concept of predicting machine failures lies in the field of manufacturing. On the one hand, we do not seek to develop a whole OC-based manufacturing architecture. On the other hand, a manufacturing plant that is controlled by an OC architecture (at least to some degree) is inevitable for evaluating our concept in this use-case. Accordingly, a suitable simulation software or framework must be chosen. For now, we decided to use a customized version of the open source manufacturing simulation framework *ManPy*[2], developed during the FP7 DREAM project [Da13]. ManPy was developed and tested using the manufacturing simulation *Plant Simulation* by Siemens, therefore its simulatory capabilities appear, at least to our needs, quite realistic. The framework itself is written in Python and uses the discrete event simulation framework *SimPy* as a basis. Changes made to ManPy up to now include the removal of unnecessary components as well as building abstraction layers for future O/C bindings.

Besides this manufacturing application scenario, we plan to identify other OC scenarios that can be used to evaluate our approach. As a manufacturing environment features more complex hardware components with (presumably) predictable physical disturbances, it appears suitable to determine a more simple scenario in order to investigate the applicability of our approach in settings where disturbances cannot be predicted. Hereby, a potential scenario might lie in the field of *Smart Home* Environments, or *Internet of Things* in general. A suitable simulation framework for such a scenario could be integrated in our existing framework by developing a wrapper using the underlying SimPy framework.

Simulation of Predictable Failures Simulating predictions made by components within the SuOC appears rather straightforward: ManPy, for example, already includes functionality to simulate machine breakdowns by repetitively sampling timestamps from a configurable population using various distributions. Based on the actual breakdown timestamp, approximate predictions can be made by e.g. slightly varying the timestamp and adding some noise. In order to evaluate the prediction capabilities located in the CM, it might be suitable to use real-world PdM datasets. However, only few freely accessible datasets for such purposes

[2] http://www.manpy-simulation.org/

can be found, provided by the UCI Machine Learning Repository [DG17] or Kaggle[3]. Besides the small number of suitable datasets, it is necessary to develop a methodology to sample realistic datapoints based on the overall state of the simulated SuOC, the control actions the CM might take, as well as potential dependencies of components within the SuOC. Furthermore, it is indeed questionable how components like jet propulsion turbines or vertical lift vehicles fit into the scope of potential future OC systems. Therefore, we tend to focus on two additional potential data sources. We plan to build various testbeds to expose rather cheap components, such as stepper motors, to different kinds of mechanical stimulation in an automated fashion. The data gathered from various sensors embedded in these testbeds might be used to determine some mathematical model to simulate the actual behaviour of the tested components under the given circumstances. Besides, approaches from mechanical engineering, such as *Stochastic Degradation*, can be used to actually simulate the actual internal behaviour of more complex components. An overview on the concept in the context of reliability engineering can be found in [Go10].

6 Conclusion

In this article, we presented a concept and a working plan for a PhD project concerned with the integration of maintenance concepts into Organic Computing Systems in order to deal with the occurrence of hardware-related failures. We gave a brief overview on how these physical disturbances can affect the robustness in existing OC systems. We formulated a problem statement focusing on the predictability of physical disturbances, the need for human repair workers as well as the scheduling of maintenance actions in OC systems. Besides, we presented a preliminary concept for a novel integrity measurement acting as an abstraction layer between the physical state of an OC system and its CM. Afterwards, we presented a working plan suggesting various approaches for solving the corresponding parts of the problem statement.

References

[AK12] Ahmad, Rosmaini; Kamaruddin, Shahrul: An overview of time-based and condition-based maintenance in industrial application. Computers & Industrial Engineering, 63(1):135–149, 2012.

[BDR13] Boudhar, H.; Dahane, M.; Rezg, N.: Spare part returns in stochastic deteriorating manufacturing system under a condition-based maintenance policy: Simulation-based Genetic Algorithm approach. IFAC Proceedings Volumes, 46(9):1399–1404, 2013.

[Da13] Dagkakis, G.; Heavey, C.; Robin, S.; Perrin, J.: ManPy: An Open-Source Layer of DES Manufacturing Objects Implemented in SimPy. In: 2013 8th EUROSIM Congress on Modelling and Simulation. pp. 357–363, Sep. 2013.

[3] http://kaggle.com

[DG17] Dua, Dheeru; Graff, Casey, University of California, Irvine, School of Information and Computer Sciences: , UCI Machine Learning Repository. http://archive.ics.uci.edu/ml, 2017.

[Fr13] Frei, Regina; McWilliam, Richard; Derrick, Benjamin; Purvis, Alan; Tiwari, Asutosh; Di Marzo Serugendo, Giovanna: Self-healing and self-repairing technologies. The International Journal of Advanced Manufacturing Technology, 69(5-8):1033–1061, 2013.

[Fu04] Fu, C.; Ye, L.; Liu, Y.; Yu, R.; Iung, B.; Cheng, Y.; Zeng, Y.: Predictive Maintenance in Intelligent-Control-Maintenance-Management System for Hydroelectric Generating Unit. IEEE Transactions on Energy Conversion, 19(1):179–186, 2004.

[Go10] Gorjian, Nima; Ma, Lin; Mittinty, Murthy; Yarlagadda, Prasad; Sun, Yong: A review on degradation models in reliability analysis. In (Kiritsis, Dimitris; Emmanouilidis, Christos; Koronios, Andy; Mathew, Joseph, eds): Engineering Asset Lifecycle Management. Springer London, London, pp. 369–384, 2010.

[GSH19] Görlich, Markus; Stein, Anthony; Hähner, Jörg: Towards Physical Disturbance Robustness in Organic Computing Systems Using MOMDPs. In: 2019 Intelligent Systems Workshop in Workshop Proceedings of the 32nd International Conference on Architecture of Computing Systems, ARCS 2019. pp. 135–142, May 2019.

[GWJ17] Guo, Yang; Wu, Zhenyu; Ji, Yang: A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction. In: 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM). IEEE, pp. 226–231, 10.08.2017 - 11.08.2017.

[IE90] : IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990, pp. 1–84, Dec 1990.

[Ja14] Janicke, Martin; Sick, Bernhard; Lukowicz, Paul; Bannach, David: Self-Adapting Multi-sensor Systems: A Concept for Self-Improvement and Self-Healing Techniques. In: 2014 IEEE SASO. IEEE, pp. 128–136, 2014.

[Kr14] Kroll, Bjorn; Schaffranek, David; Schriegel, Sebastian; Niggemann, Oliver: System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants. In: IEEE [International Conference on] Emerging Technologies and Factory Automation (ETFA), 2014. IEEE, Piscataway, NJ, pp. 1–7, 2014.

[KS17] Kanawaday, Ameeth; Sane, Aditya: Machine learning for predictive maintenance of industrial machines using IoT sensor data. In: 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, pp. 87–90, 24.11.2017 - 26.11.2017.

[Le11] Lewis, P. R.; Chandra, A.; Parsons, S.; Robinson, E.; Glette, K.; Bahsoon, R.; Torresen, J.; Yao, X.: A Survey of Self-Awareness and Its Application in Computing Systems. In: 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops. pp. 102–107, Oct 2011.

[LMD13] Lalanda, Philippe; McCann, Julie A.; Diaconescu, Ada: Autonomic Computing. Springer London, London, 2013.

[LVT16] Ladj, A.; Varnier, C.; Tayeb, F. Benbouzid-Si: IPro-GA: an integrated prognostic based GA for scheduling jobs and predictive maintenance in a single multifunctional machine. IFAC-PapersOnLine, 49(12):1821–1826, 2016.

[Ma11] Maehle, Erik; Brockmann, Werner; Grosspietsch, Karl-Erwin; El Sayed Auf, Adam; Jakimovski, Bojan; Krannich, Stephan; Litza, Marek; Maas, Raphael; Al-Homsy, Ahmad: Application of the Organic Robot Control Architecture ORCA to the Six-Legged Walking Robot OSCAR. In: Organic Computing — A Paradigm Shift for Complex Systems, pp. 517–530. Springer Basel, Basel, 2011.

[MST17a] Müller-Schloer, Christian; Tomforde, Sven: Building Organic Computing Systems. In (Müller-Schloer, Christian; Tomforde, Sven, eds): Organic Computing – Technical Systems for Survival in the Real World, pp. 171–258. Birkhäuser, Cham, 2017.

[MST17b] Müller-Schloer, Christian; Tomforde, Sven: Organic Computing – Technical Systems for Survival in the Real World. Birkhäuser, Cham, 2017.

[Na09] Nafz, Florian; Ortmeier, Frank; Seebach, Hella; Steghofer, Jan-Philipp; Reif, Wolfgang: A generic software framework for role-based Organic Computing systems. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2009. IEEE, Piscataway, NJ, pp. 96–105, 2009.

[NDG14] Nguyen, Kim-Anh; Do, Phuc; Grall, Antoine: Condition-based maintenance for multi-component systems using importance measure and predictive information. International Journal of Systems Science: Operations & Logistics, 1(4):228–245, 2014.

[On10] Ong, Sylvie C. W.; Png, Shao Wei; Hsu, David; Lee, Wee Sun: Planning under Uncertainty for Robotic Tasks with Mixed Observability. The International Journal of Robotics Research, 29(8):1053–1068, 2010.

[PDZ10] Peng, Ying; Dong, Ming; Zuo, Ming Jian: Current status of machine prognostics in condition-based maintenance: a review. The International Journal of Advanced Manufacturing Technology, 50(1-4):297–313, 2010.

[Pr11] Prothmann, Holger and Tomforde, Sven and Branke, Jürgen and Hähner, Jörg and Müller-Schloer, Christian and Schmeck, Hartmut: Organic Traffic Control. In (Müller-Schloer, Christian and Schmeck, Hartmut and Ungerer, Theo, ed.): Organic Computing — A Paradigm Shift for Complex Systems, pp. 431–446. Springer Basel, Basel, 2011.

[Sc10] Schmeck, Hartmut; Müller-Schloer, Christian; Çakar, Emre; Mnif, Moez; Richter, Urban: Adaptivity and self-organization in organic computing systems. ACM TAAS, 5(3):1–32, 2010.

[Sc11] Schmitt, Julia; Roth, Michael; Kiefhaber, Rolf; Kluge, Florian; Ungerer, Theo: Using an Automated Planner to Control an Organic Middleware. In: 2011 IEEE SASO. IEEE, pp. 71–78, 2011.

[Se17] Selcuk, Sule: Predictive maintenance, its implementation and latest trends. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 231(9):1670–1679, 2017.

[ST16] Sommer, Matthias; Tomforde, Sven: Concepts for Resilient Traffic Management based on Organic Computing. Technical Report 2016-06, University of Augsburg, Faculty of Computer Science, 2016.

[St17] Stein, Anthony: Reaction Learning. In: Organic Computing – Technical Systems for Survival in the Real World, chapter Basic Methods, pp. 287–328. Birkhäuser, Cham, 2017.

[STH16] Sommer, Matthias; Tomforde, Sven; Hähner, Jörg: An Organic Computing Approach to Resilient Traffic Management. In: Autonomic Road Transport Support Systems, pp. 113–130. Springer, Cham, 2016.

[To11] Tomforde, Sven; Prothmann, Holger; Branke, Jürgen; Hähner, Jörg; Mnif, Moez; Müller-Schloer, Christian; Richter, Urban; Schmeck, Hartmut: Observation and Control of Organic Systems. In (Müller-Schloer, Christian; Schmeck, Hartmut; Ungerer, Theo, eds): Organic Computing — A Paradigm Shift for Complex Systems, pp. 325–338. Springer Basel, Basel, 2011.

[To18] Tomforde, Sven; Kantert, Jan; Müller-Schloer, Christian; Bödelt, Sebastian; Sick, Bernhard: Comparing the Effects of Disturbances in Self-adaptive Systems - A Generalised Approach for the Quantification of Robustness. In: Transactions on Computational Collective Intelligence XXVIII, volume 10780 of LNCS, pp. 193–220. Springer, Cham, 2018.

[vP13] van Horenbeek, Adriaan; Pintelon, Liliane: A dynamic predictive maintenance policy for complex multi-component systems. Reliability Engineering & System Safety, 120:39–50, 2013.

[Wu07] Wu, Sze-jung; Gebraeel, Nagi; Lawley, Mark A.; Yih, Yuehwern: A Neural Network Integrated Decision Support System for Condition-Based Optimal Predictive Maintenance Policy. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 37(2):226–236, 2007.

[Zi11] Ziermann, Tobias and Wildermann, Stefan and Teich, Jürgen: OrganicBus: Organic Self-organising Bus-Based Communication Systems. In (Müller-Schloer, Christian and Schmeck, Hartmut and Ungerer, Theo, ed.): Organic Computing — A Paradigm Shift for Complex Systems, pp. 489–501. Springer Basel, Basel, 2011.